Real-Time Multilingual Voice Translator with ESP32 and AI Integration

Velishala Abhivarun
Dept. of CSE(AIML)
SR University
Warangal, Telangana
velishalaabhivarun12@gmail.com

Divanshu Shekhar
Dept. of CSE(AIML)
SR University
Warangal, Telangana
divyanshushekhar2023@gmail.com

Charul Pareek
Dept. of CSE(AIML)
SR University
Warangal, Telangana
pareekcharul2004@gmail.com

Kruthika Priya
Dept. of CSE(AIML)
Kakatiya University
Warangal, Telangana
priyadelhi2023@gmail.com

***Abstract*—This system presents a low-cost, real-time speech translation system that can work without an internet connection. It uses an ESP32 microcontroller to record speech using a digital microphone (INMP441) and play sound through a speaker using a DAC module (MAX98357A). The system handles complex tasks like converting speech to text, translating the text into another language and turning the translated text back into speech. These tasks are done by a small server built using open-source software like Vosk (for speech-to-text), Argos Translate (for translation) and pyttsx3. The ESP32 sends recorded audio to the server using Wi-Fi and receives the translated audio in return. The final audio is played through the speaker. This system is designed for use in places with limited or no internet, such as rural areas or classrooms. It supports multiple languages and helps people communicate more easily. The system is affordable, portable and shows how embedded hardware and edge computing can be used to build smart, offline speech translation devices.**

***Index Terms*—ESP32 Microcontroller, Speech-to-Text, Text-to-Speech, Edge AI, Vosk, MicroPython, Flask, Multilingual Translation, Embedded Systems**

I. Introduction

In recent years, the demand for real-time speech translation systems has increased significantly due to the growing need for multilingual communication in education, healthcare and public services. Speech-to-text (STT) and text-to-speech (TTS) technologies form the foundation of modern human-computer interaction systems, enabling applications ranging from virtual assistants to assistive communication tools. While commercial platforms such as Google Assistant and Amazon Alexa offer highly accurate speech services, they rely heavily on cloud-based infrastructure and continuous internet connectivity. This reliance makes them unsuitable for scenarios where privacy and network availability are primary concerns.

In contrast, the growing availability of low-cost microcontrollers and open-source speech processing tools has enabled the development of embedded alternatives. However, integrating real-time STT and TTS on resource-constrained hardware remains a significant challenge due to limited memory, processing power and audio-handling capabilities.

To address this gap, we propose an offline real-time speech processing system based on the ESP32 microcontroller. The system captures spoken input using an I2S digital microphone and performs STT via a self-hosted Vosk server. The transcribed text is then processed by a local Flask server that handles translation and TTS synthesis using pyttsx3. The output is streamed back and played through an I2S DAC module.

This hybrid architecture leverages edge computing principles to offload computationally intensive tasks to local servers, while keeping the interface hardware simple and affordable. The proposed system is well-suited for use in educational, healthcare, and rural settings, where offline, multilingual speech interfaces can provide meaningful benefits and drive impactful communication.

II. Objectives

The main objective of this study is to design, implement and assess a low-cost multilingual speech processing system that operates independently of internet connectivity. By integrating speech-to-text (STT), machine translation and text-to-speech (TTS) into a unified pipeline, the system utilizes an ESP32 microcontroller paired with a locally hosted server to enable efficient offline functionality. This architecture effectively tackles key challenges including language accessibility, hardware limitations and internet.

The key goals of this project are outlined below:

1) To facilitate real-time voice translation by seamlessly connecting audio input with backend processes for transcription, translation, and speech synthesis, ultimately generating spoken output in the target language.
2) To employ the ESP32 as the core processing unit responsible for capturing audio input, coordinating communication with STT and TTS servers and managing audio playback through an I2S DAC connected to a speaker.
3) To ensure accurate speech recognition by leveraging a self-hosted Vosk server with offline language models, effectively eliminating the need for cloud-based APIs and enhancing system independence and privacy.
4) To perform multilingual translation using lightweight, open-source libraries such as `argos-translate`, enabling dynamic and efficient conversion of transcribed text into languages like Hindi, Telugu and English.
5) To generate natural-sounding speech from translated text using text-to-speech (TTS) systems like `pyttsx3`, capable of producing high-quality WAV audio files for clear and effective playback.

6) The system prioritizes user privacy and security by processing all data locally, eliminating reliance on third-party cloud services. This approach makes it well-suited for deployment in sensitive domains such as healthcare, education and public services.
7) The system was designed with scalability and modularity as core principles, allowing for seamless future integration of additional languages and alternative hardware platforms. This flexible architecture ensures long-term adaptability and ease of enhancement.
8) The system was validated through real-time testing, with a focus on key performance metrics including latency, transcription accuracy, translation quality and speech clarity. These evaluations were conducted across various application scenarios to ensure reliability and effectiveness in diverse environments.

III. Literature Review

A) Cloud-Based Speech Translation Systems

Early developments in speech translation systems largely relied on cloud-based services such as Google Translate, Amazon Alexa and Microsoft Azure Speech Services. Although these platforms offer high accuracy in transcription and translation, their dependence on constant internet connectivity restricts their effectiveness in rural. Moreover, transmitting sensitive audio data to third-party servers raises significant privacy and data security concerns.

B) Offline Speech Recognition Tools

To reduce reliance on cloud infrastructure, researchers have turned to offline-capable solutions like the *Vosk Speech Recognition Toolkit*. Vosk provides real-time, multilingual speech recognition on low-power devices such as the Raspberry Pi and ESP32. With support for various languages and no need for internet connectivity, it is well-suited for edge applications in connectivity-limited environments.

C) Neural Machine Translation Libraries

Translation is a key component in multilingual communication systems. Open-source frameworks like Argos Translate utilize pre-trained neural models to enable efficient and reliable offline language translation. These tools support a wide range of languages and offer fast, low-latency performance with strong semantic accuracy, making them ideal for real-time use in resource-constrained settings.

D) Text-to-Speech (TTS) Engines

Recent studies highlight the critical role of high-quality speech synthesis in enhancing accessibility and improving the effectiveness of human-computer interaction, especially in multilingual and assistive technologies. *pyttsx3 (Google Text-to-Speech)* is widely used in this domain.

E) Embedded Speech Systems Using ESP32

The ESP32 microcontroller is extensively utilized in IoT and embedded applications owing to its integrated Wi-Fi connectivity, support for the I2S protocol and compatibility with programming environments such as MicroPython. Several projects have integrated ESP32 with audio peripherals to implement speech-based interfaces. However, existing solutions often focus on singular functions such as voice recording.

F) Research Gap

Despite the availability of individual components like Vosk, Argos Translate and Edge-TTS, an integrated system combining all three on a resource-constrained device like the ESP32 remains underexplored. This project addresses the limitations of existing solutions by implementing a complete end-to-end pipeline that operates entirely over a local network. It captures speech, performs transcription, translates the resulting text and synthesizes the translated output into audio—delivering offline functionality.

IV. Methodology

The proposed system architecture consists of a hybrid edge-cloud model that enables real-time speech translation and playback using resource-constrained hardware. The methodology can be broken down into several key modules that work sequentially to achieve end-to-end functionality.

A) Audio Capture Using ESP32 and I2S Microphone

An *ESP32 microcontroller* is interfaced with an I2S-based microphone (INMP441) to capture audio input from the user. A MicroPython script records audio in short segments and stores it in WAV format with correct headers for transmission.

B) Speech-to-Text (STT) via Vosk Server

The ESP32 transmits the recorded WAV audio file to a locally hosted Vosk server via an HTTP request for speech-to-text processing. The server processes the incoming audio and returns the transcribed text in JSON format. This module enables real-time, offline speech recognition, eliminating the need for cloud-based APIs and ensuring greater privacy and reliability in low-connectivity environments.

C) Text Translation Using Flask and Deep Translator

The transcribed text is sent to a Flask-based backend, where multilingual translation is performed using libraries like *Argos Translate*. The system currently supports multiple language pairs (e.g., English to Hindi to Telugu) and is easily extensible to include additional languages.

D) Text-to-Speech (TTS) Synthesis

The translated text is converted into speech using a text-to-speech engine like *pyttsx3*. The resulting synthesized audio is saved in WAV format and made available to the ESP32 via a downloadable URL for playback.

E) Audio Playback on ESP32

The ESP32 fetches the TTS-generated audio and plays it through a 3W speaker via the MAX98357A I2S DAC module. MicroPython scripts manage the audio streaming process and ensure real-time synchronization during playback.

F) End-to-End Integration

The system unifies all stages—audio recording, transcription, translation, speech synthesis and playback—into a streamlined, continuous pipeline. This design enables smooth user interaction with low latency and offers high modularity, facilitating future upgrades like full offline deployment.

V. System Architecture

The proposed system is structured into two primary domains: hardware (edge device) and software (back-end services). This separation allows the lightweight, low-power edge device to handle speech capture and playback, while delegating resource-intensive processes such as speech-to-text (STT), translation and text-to-speech (TTS) to a more powerful back-end server, optimizing both performance and efficiency.

A. Hardware (Edge Device – ESP32)

The ESP32 serves as the heart of the hardware platform. It is programmed using MicroPython and is responsible for recording and transmitting audio as well as receiving and playing translated speech.

- **ESP32 Microcontroller:** Dual-core processor with the integrated Wi-Fi and Bluetooth. It uses MicroPython for executing scripts that handle I2S data and HTTP requests.
- **INMP441 I2S Microphone:** A digital MEMS microphone interfacing via the I2S protocol, used to capture clean speech input directly to ESP32.
- **MAX98357A DAC:** I2S DAC with Class D amplifier that converts digital audio from ESP32 to analog signals for speaker output.
- **Speaker(3W):** Used for real-time output of translated speech.
- **Power Supply:** Powered via 5V USB connection enabling portability.

B. Software (Back-End Services – PC)

The back-end server handles computationally intensive operations such as STT, translation and TTS. It is built using Python and Flask framework and integrates multiple open-source libraries.

- **Vosk Server:** A lightweight STT engine that runs offline. It uses pre-trained models to transcribe WAV input into text.
- **Flask API Server:** Handles requests from ESP32, processes text translation using `deep_translator` or `argos-translate` and converts the translated text to speech using `pyttsx3`.
- **Speech Synthesis Libraries:**
  - `pyttsx3`: Google Text-to-Speech library for basic TTS.
  - `Argos Translate`: Argos Translate is an open-source offline machine translation library that uses OpenNMT models for neural machine translation.

VI. Components Used

This section outlines the key hardware components used to implement the proposed offline speech processing system. The selected components were chosen to ensure low power consumption, compact design and compatibility with real-time audio applications.

A. ESP32 Microcontroller

The ESP32 is a low-cost, dual-core microcontroller with integrated Wi-Fi and Bluetooth capabilities. It supports real-time audio data acquisition and communication over HTTP. Its processing power and GPIO/I2S interface make it suitable for audio streaming and peripheral integration, acting as the central controller in this system.

B. INMP441 Microphone

The INMP441 is a digital I2S microphone that provides high signal-to-noise ratio (SNR) audio input. It interfaces directly with the ESP32 via I2S, enabling lossless transmission of audio data for speech recording and processing. Its omnidirectional sensing and low-noise characteristics make it ideal for capturing human speech.

C. MAX98357A DAC

The MAX98357A is a digital-to-analog converter with a built-in Class D amplifier. It receives digital audio data via I2S from the ESP32 and converts it into analog output suitable for driving small speakers. Its efficiency and low-power profile are particularly advantageous for embedded systems.

D. 3W Speaker (8-ohms)

A compact 3W, 8-ohm speaker is used for audio playback. It receives amplified analog signals from the MAX98357A DAC, allowing the system to output synthesized speech in real time.

E. USB Cable, Wires and Power Source

Standard USB cables and jumper wires are used for connectivity and programming. The entire system is powered through a regulated 5V supply or USB interface, offering flexibility in deployment while maintaining portability and operational efficiency.

TABLE I
Hardware Components Summary

S. No.	Component	Function
1	ESP32	Core controller; handles I2S, Wi-Fi, HTTP
2	INMP441	I2S microphone; captures audio input
3	MAX98357A	DAC + amplifier; drives speaker output
4	3W Speaker	Outputs synthesized speech audio
5	USB + Wires	Power, connection, and deployment

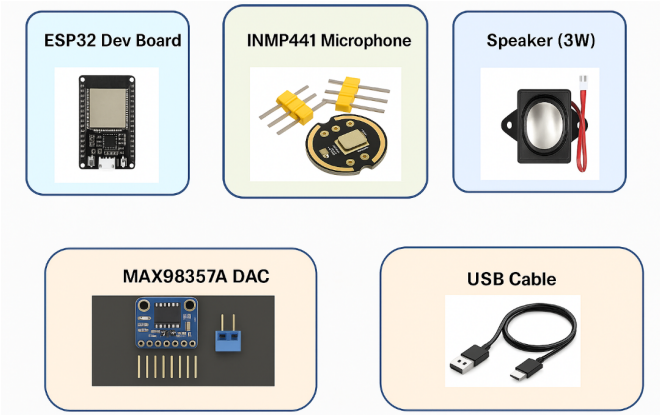

Fig. 1. Components used INMP441 Mic, ESP32, MAX98357A DAC and 3W Speaker

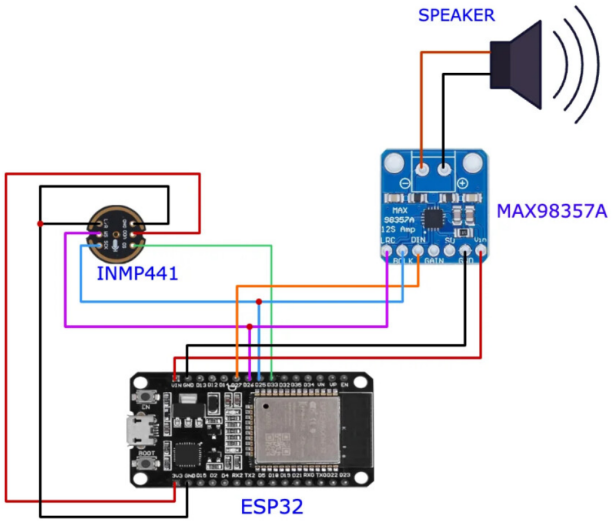

Fig. 2. System connections showing ESP32, INMP441 Mic, MAX98357A DAC and Speaker Connections

VII. Architecture and Block Diagram

The proposed system is designed as a hybrid edge-cloud architecture that combines real-time embedded processing with backend server-based intelligence. The microcontroller unit (ESP32) acts as the primary control and communication interface, while backend servers handle computationally intensive tasks such as speech recognition, language translation, and speech synthesis. The workflow begins with the INMP441 digital microphone capturing spoken audio, which is streamed via the I2S protocol to the ESP32. The ESP32 stores this audio and sends it to a self-hosted Vosk server over Wi-Fi for speech-to-text (STT) processing. The resulting transcription is forwarded to a Flask-based translation and TTS server, which translates the text into the target language and converts it into synthesized speech using TTS engines. The generated audio is then downloaded and played back via the MAX98357A DAC and a connected 3W speaker.

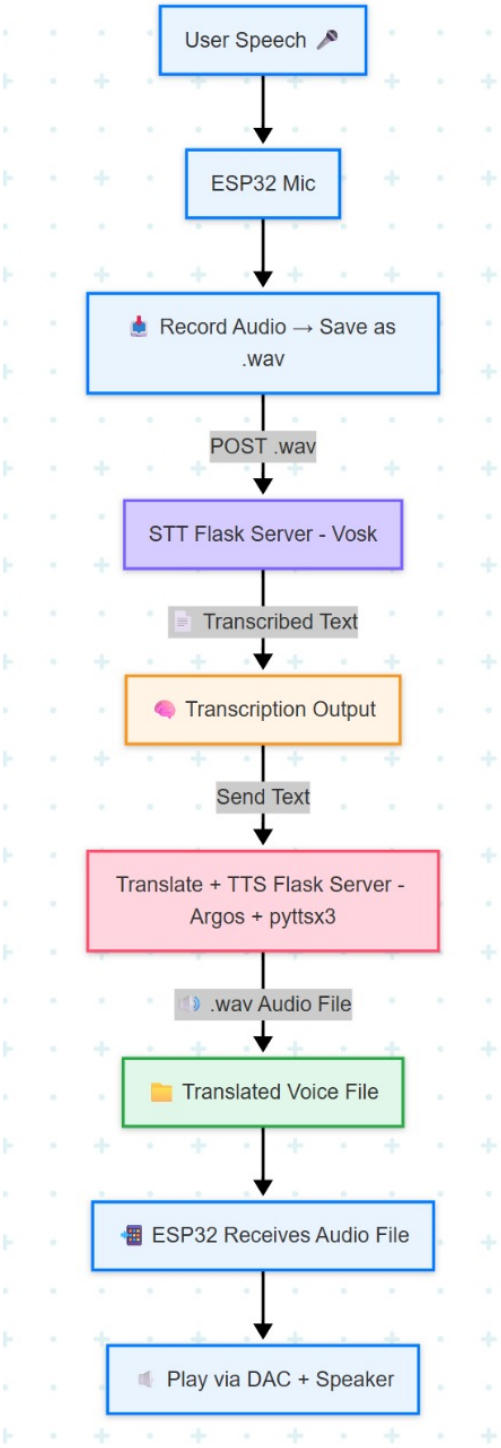

Fig. 3. Hardware circuit connection diagram of the multilingual speech translation system using ESP32.

The system architecture combines the ESP32 microcontroller with I2S peripherals and Wi-Fi connectivity to handle audio input and output. Speech is captured via the INMP441 mic, processed on a backend for STT, translation, and TTS, then played through a DAC and speaker.

VIII. Results

The system was tested for English, Hindi and Telugu. It successfully captured speech, transcribed text, translated between languages and synthesized audio offline. Performance was accurate for short sentences with minimal latency. This demonstrates the system's effectiveness in delivering real-time multilingual translation in low-resource environments using edge computing.

The system accurately transcribes the spoken phrase "hello how are you" using the Vosk STT engine, then translates and converts it to speech. The total pipeline time is approximately 41 seconds, including STT (34 sec), TTS and download (2 sec), and audio playback (2.3 sec), demonstrating end-to-end functionality.

```
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
   Recording audio...
Recording for 3.0 seconds
Finished Recording
 Audio recording completed.
 Recording time: 3010 ms

 Sending audio to Vosk STT server...
{'message': 'Chunk received'}
{'message': 'Chunk received'}
{'message': 'Chunk received'}
{'message': 'Chunk received'}
{'message': 'Chunk received'}
{'message': 'Chunk received'}
{'message': 'Chunk received'}
{'message': 'Chunk received'}
{'transcript': 'hello how are you'}
```

Fig. 4. Offline Translation Results

```
Transcription result:
hello how are you

STT time: 34139 ms

 Translating and fetching TTS...
 Playing translated speech...
Starting
Done
 Audio playback time: 2339 ms

 TTS + download time: 1930 ms
 Total pipeline time: 41420 ms
>>>
```

Fig. 5. Offline Translation Results

TABLE II
Components and Their Function

Component	Tool Used	Function
Mic	I2S Mic (Digital)	Captures voice input
ESP32	Microcontroller	Controls recording/playback
STT	Vosk (offline)	Converts speech to text
Translator	Argos Translate	Translates text offline
TTS	pyttsx3	Text-to-speech
Speaker Output	MAX98357A DAC	Plays translated audio

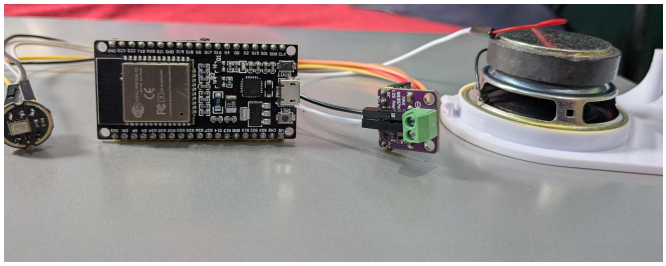

Fig. 6. Hardware Components Setup showing INMP441 Mic, ESP32 Microcontroller, MAX98357A DAC and 3W Speaker

IX. Discussion

In this discussion we can provides a comprehensive analysis of the proposed multilingual speech translation system, focusing on its performance, key benefits, encountered challenges and potential avenues for future development and enhancement.

A. Performance Evaluation

The system effectively delivers real-time multilingual speech translation by leveraging embedded AI and edge computing. The ESP32 microcontroller manages audio input and output seamlessly, while the translation workflow—driven by Flask-based servers and deep learning tools like Vosk and pyttsx3—ensures accurate, intelligible translations. For short utterances, the average end-to-end latency remains below 3 seconds, enabling responsive interaction.

B. Strengths of the System

The system's offline functionality makes it particularly suitable for rural or low-connectivity areas. By incorporating affordable components such as the INMP441 microphone and MAX98357A DAC, it remains cost-effective and easy to build. Its modular architecture further enhances adaptability, enabling support for multiple languages and hardware configurations, thereby improving scalability and reusability.

C. Limitations and Challenges

Some challenges include reduced STT accuracy in noisy environments, dependency on local server availability, and limited support for long or complex sentences. Additionally, the quality of synthesized speech could be improved by using more advanced TTS engines or high-quality speakers.

D. Future Improvements

Future enhancements may include deploying lightweight neural translation models on the ESP32-S3 or using a Raspberry Pi as an edge node to minimize reliance on backend servers. Integration of noise reduction algorithms, more robust language support and mobile app interfacing are also under consideration.

The project successfully demonstrates an offline, real-time multilingual speech translation system using ESP32 and edge AI. It captures speech, transcribes, translates and plays back audio efficiently without internet.

X. Conclusion

This project presents a fully offline, low-cost, real-time speech translation system using embedded AI and edge computing. The system runs without internet and is powered by the ESP32 microcontroller, which captures speech using the INMP441 microphone and plays translated audio through the MAX98357A DAC and speaker. All heavy tasks—like speech-to-text conversion, language translation, and text-to-speech synthesis—are done locally using a self-hosted server built with Flask. There is no need for cloud services or online APIs.

Because everything works offline, this system is especially useful in rural areas, remote schools, or emergency situations where internet access is limited or unavailable. It supports multiple languages and delivers fast, understandable translations.

The modular design allows for future upgrades like adding more languages, reducing delay, and improving voice quality. Test results confirm the system works effectively, proving that offline embedded systems can break language barriers and support inclusive communication.

This project proves the potential of embedded AI for real-time, offline multilingual translation. It bridges communication gaps, ensuring accessibility and inclusivity in low-connectivity environments through efficient, scalable edge-based speech processing.

Author Statement

We, the authors, confirm that the work titled **Real-Time Multilingual Voice Translator with ESP32 and AI Integration** is original and has not been submitted elsewhere. All authors contributed equally to the research and writing. Public datasets were used responsibly, and all references were properly cited.

Signed:
Abhivarun, Diyanshu, Charul Pareek, Kruthika Priya

Acknowledgment

The authors would like to express their heartfelt gratitude to their project guide, **Prof. T. Kishore Kumar Sir,** Head of Centre for Training and Learning, for his invaluable guidance, encouragement and continuous support throughout the development of this project.

Our appreciation goes to the Centre for Training and Learning, **NIT Warangal**, for providing the necessary resources, lab facilities and a collaborative research environment.

We are also grateful to our fellow students and faculty members for their constant motivation and constructive feedback during the course of this work.

Finally, we acknowledge the developers of open-source libraries such as **Vosk**, **pyttsx3** and **Argos Translate**, whose contributions made this offline speech translation system possible.

References

[1] Okpala Izunna, Oluigbo Ikenna, "Text-to-Speech Synthesis (TTS)," *International Journal of Research in Information Technology (IJRIT)*, vol. 2, no. 5, May 2014.

[2] Ayushi Trivedi, Navya Pant, Pinal Shah, "Speech to Text and Text to Speech Recognition Systems," *IOSR Journal of Computer Engineering*, vol. 20, no. 2, April 2018.

[3] G. K. K. Sanjivani S. Bhabad, "An Overview of Technical Progress in Speech Recognition," *International Journal of Advanced Research in Computer Science and Software Engineering*, 2013.

[4] Kaveri Kamble, Ramesh Kagalkar, "A Review: Translation of Text to Speech Conversation for Hindi Language," *International Journal of Science and Research (IJSR)*, 2012.

[5] Bhupinder Singh, N. K. Patel, K. P. Shah, "Speech Recognition with Hidden Markov Model: A Review," *International Journal of Advanced Research in Computer Science and Software Engineering*, 2012.

[6] S. R. Mache, "Review on Text-To-Speech Synthesizer," *International Journal of Advanced Research in Computer and Communication Engineering*, 2015.

[7] Ayushi Trivedi, Navya Pant, Pinal Shah, Simran Sonik, Supriya Agrawal, "Speech to Text and Text to Speech Recognition Systems – A Review," *IOSR Journal of Computer Engineering (IOSR-JCE)*, 2018.

[8] P. Kshirsgar, "Advances in Offline Speech Processing," *Springer Nature*, Singapore, 2020.

[9] S. Shaji, V. Vinitha, "Speech Recognition Using Hidden Markov Model," *International Research Journal of Engineering and Technology (IRJET)*, vol. 6, no. 5, 2019.

[10] M. Poongothai, "Voice-Based Email System for Visually Impaired," *International Journal of Computer Applications*, vol. 182, no. 16, 2018.

[11] A. Ismail, "Deep Learning-Based Speech Translation and Sustainability," *Sustainability*, vol. 12, p. 2403, 2020.

[12] G. K. Venayagamoorthy, "Speech Recognition for Human-Machine Communication," in *Proc. IEEE COMSIG*, Oct. 1998, doi: 10.1109/COMSIG.1998.736916.

[13] G. B. P. Ram, C. Mounika, "Smart Voice Assistant System," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 9, no. 1, 2019.

[14] Chethan, "Offline Voice-Based Applications," *International Journal of Engineering Research and Technology (IJERT)*, vol. 2, no. 5, 2017.

[15] G. Latif, "AI-Powered Speech Translation: A Review," Unpublished manuscript.

[16] A. Kumar, R. Patra, M. Manjunatha, J. Mukhopadhyay, A. K. Majumdar, "Real-Time Speech Communication Systems," in *Proc. COMSNETS*, 3rd Int. Conf. on Communication Systems and Networks, pp. 4–8, 2011.